|  |
| --- |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  **TRƯỜNG ĐẠI HỌC HOA SEN**  **KHOA HỌC VÀ CÔNG NGHỆ** |

**Đề tài:**

**Giảng viên hướng dẫn :** Võ Hoàng Hải

**Lớp :**

**Nhóm sinh viên thực hiện :** Phan Quốc Anh - 091371

**:** Nguyễn Mạnh Hoàng - 091398

**:** Nguyễn Đình Thi - 091455

**THÁNG 11 / NĂM 2011**

|  |
| --- |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  **TRƯỜNG ĐẠI HỌC HOA SEN**  **KHOA HỌC VÀ CÔNG NGHỆ** |

**Đề tài:**

**Giảng viên hướng dẫn :** Võ Hoàng Hải

**Lớp :**

**Nhóm sinh viên thực hiện :** Phan Quốc Anh - 091371

**:** Nguyễn Mạnh Hoàng - 091398

**:** Nguyễn Đình Thi - 091455

**THÁNG 11 / NĂM 2011**

# LỜI MỞ ĐẦU

Mục đích:Người dùng có thể tìm kiếm thông tin địa điểm, đường đi và cho phép người dùng lưu trữ những địa điểm yêu thích của họ.

Các chức năng của đề tài:

* Đăng nhập
* Liệt kê các địa điểm thường xảy ra kẹt xe
* Tìm kiếm thông tin một địa điểm cụ thể
* Tìm kiếm đường đi từ điểm A tới điểm B
* Tìm kiếm đường đi từ điểm A tới điểm B theo phương tiện di chuyển (xe máy, đường bộ, máy bay, …)
* Tìm đường đi ngắn nhất giữa hai địa điểm
* Danh sách địa điểm yêu thích

Danh sách sinh viên thực hiện đề tài:

|  |  |  |
| --- | --- | --- |
| **Họ tên** | **MSSV** | **Email** |
| Phan Quốc Anh | 091371 | anh.pq1371@sinhvien.hoasen.edu.vn |
| Nguyễn Mạnh Hoàng | 091398 | hoang.nm1398@sinhvien.hoasen.edu.vn |
| Nguyễn Đình Thi | 091455 | thi.nd1455@sinhvien.hoasen.edu.vn |

# NHẬN XÉT

**Giảng viên nhận xét kí tên**

# MỤC LỤC

[LỜI MỞ ĐẦU 1](#_Toc309767510)

[NHẬN XÉT 2](#_Toc309767511)

[MỤC LỤC 3](#_Toc309767512)

[LỜI CẢM ƠN 5](#_Toc309767513)

[**1.** **LÝ THUYẾT** 6](#_Toc309767514)

[**1.1.** **Webservice là gì?** 6](#_Toc309767515)

[**1.2.** **SOA là gì** 6](#_Toc309767516)

[**1.3.** **Đặc điểm của hướng dịch vụ:** 7](#_Toc309767517)

[**1.4.** **Tổng quan SOA** 7](#_Toc309767518)

[**1.5.** **Các bước của cách tiếp cận SOA** 8](#_Toc309767519)

[**1.6.** **Service Contract** 9](#_Toc309767520)

[**1.7.** **Service Coupling** 9](#_Toc309767521)

[**1.8.** **Service Abstraction:** 9](#_Toc309767522)

[**1.8.1.** **Types of Meta Abstraction** 9](#_Toc309767523)

[**1.8.2.** **Technology Information Abstraction** 10](#_Toc309767524)

[**1.8.3.** **Functional Abstraction** 10](#_Toc309767525)

[**1.8.4.** **Programmatic Logic Abstraction** 10](#_Toc309767526)

[**1.8.5.** **Quality of Service Abstraction** 10](#_Toc309767527)

[**1.8.6.** **Service Abstraction and Service Design** 10](#_Toc309767528)

[**1.8.7.** **Service Abstraction and Standardized Service Contract** 11](#_Toc309767529)

[**1.9.** **Service Reusability** 11](#_Toc309767530)

[**1.10.** **Service Autonomy:** 11](#_Toc309767531)

[**1.11.** **Service Statelessness** 13](#_Toc309767532)

[**1.12.** **Service Discoverability** 13](#_Toc309767533)

[**1.13.** **Service Composability** 13](#_Toc309767534)

[**2.** **THIẾT KẾ CHƯƠNG TRÌNH** 14](#_Toc309767535)

[**2.1.** **Đặc tả các chức năng** 14](#_Toc309767536)

[**2.1.1.** **Đăng ký** 14](#_Toc309767537)

[**2.1.2.** **Đăng nhập** 14](#_Toc309767538)

[**2.1.3.** **Đăng xuất** 14](#_Toc309767539)

[**2.1.4.** **Danh sách địa điểm (đường) cần lưu ý** 15](#_Toc309767540)

[**2.1.5.** **Tìm kiếm thông tin đường** 15](#_Toc309767541)

[**2.1.6.** **Tìm kiếm thông tin địa điểm** 16](#_Toc309767542)

[**2.1.7.** **Tìm kiếm đường đi từ điểm A tới điểm B** 17](#_Toc309767543)

[**2.1.8.** **Tìm đường đi ngắn nhất giữa hai địa điểm** 18](#_Toc309767544)

[**2.1.9.** **Danh sách địa điểm yêu thích** 18](#_Toc309767545)

[**2.1.10.** **Đánh giá địa điểm (Rating)** 19](#_Toc309767546)

[**2.1.11.** **Danh sách các địa điểm yêu thích nhất** 19](#_Toc309767547)

[**2.2.** **Cơ sở dữ liệu** 20](#_Toc309767548)

[**3.** **HIỆN THỰC** 21](#_Toc309767549)

[KẾT LUẬN 22](#_Toc309767550)

[TÀI LIỆU THAM KHẢO 23](#_Toc309767551)

# LỜI CẢM ƠN

1. **LÝ THUYẾT**
   1. **Webservice là gì?**

* Web services là những application component
* Web services được khám phá bởi sử dụng UDDI (Universal Description, Discovery and Integration)
* Web services được sử dụng bởi các ứng dụng khác
* Ngôn ngữ cơ bản được sử dụng cho WS là XML

Hoạt động WS:

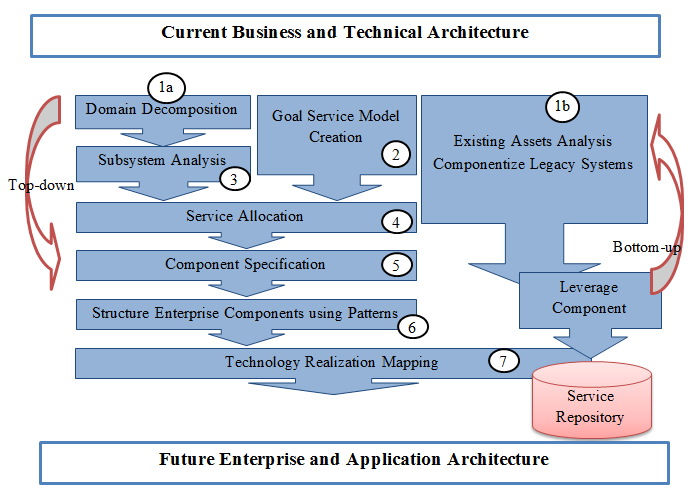
* WS được sử dụng giữa các flatform và ngôn ngữ lập trình khác nhau
* Giao thức phổ biến được dùng là HTTP
  1. **SOA là gì**
* Viết tắt từ Service Oriented Architecture
* Theo DotNetGuru : SOA là khái niệm về hệ thống mà trong đó mỗi ứng dụng được xem là một nguồn cung cấp dịch vụ
* SOA là tập hợp các dịch vụ kết nối 'mềm dẻo' với nhau, có giao tiếp được định nghĩa rõ ràng và độc lập với nền tảng hệ thống, và có thể tái sử dụng
* SOA là cấp độ cao hơn của phát triển ứng dụng.
* SOA thiết kế tách riêng phần thực hiện dịch vụ và giao tiếp gọi dịch vụ

|  |  |
| --- | --- |
| **SOA** | **Web Service** |
| SOA là **kiến trúc phần mềm** phát xuất từ:  + định nghĩa giao tiếp  + xây dựng toàn bộ mô hình **ứng dụng như là mô hình các giao tiếp** | - WS được truy cập thông qua giao tiếp, thường theo cách thức yêu cầu - đáp trả (request - response)  - Dịch vụ web là đặc tả công nghệ   * + WS dựa trên các nền tảng: WSDL, SOAP, UDDI cho phép xây dựng các giải pháp lập trình cho các ứng dụng   + Hiện nay công nghệ này có thể hoàn thiện hơn hay đã thay thế bằng công nghệ khác tốt hơn ( mặc dù công nghệ vẫn tốt) |
| Một tương tác định hướng dịch vụ luôn bao hàm **một cặp đối tác: nguồn cung cấp dịch vụ và khách hàng sử dụng dịch vụ**. ( quy mô lớn hơn chứ không đơn thuần là bên là đưa ra 1 request và 1 bên là response như WS ) | Với yêu cầu dịch vụ 1 chiều thì WS vẫn là yêu cầu trực tiếp có chủ đích từ **một phần mềm này đến một phần mềm khác** |
| SOA cũng có thể thực hiện với các giải pháp kỹ thuật khác không phải dịch vụ web (và không phải tất cả dịch vụ web đều có kiến trúc SOA) | Dịch vụ web đưa ra giải pháp kỹ thuật để thực hiện SOA |

* 1. **Đặc điểm của hướng dịch vụ:**
* Tăng tính nhất quán việc trình bày dữ liệu và chức năng
* Giảm sự phụ thuộc giữa các đơn vị xử lý giải pháp
* Ẩn những thông tin bên dưới của thiết kế và chi tiết thực thi
* Tăng khả năng sử dụng lại cho nhiều mục đích xử lý
* Tăng cơ hội kết hợp các xử lý vào những cấu hình khác nhau
* Tăng tính dự đoán
* Có khả năng mở rộng và tính sẵn có
* Tăng khả năng nhận biết xử lý giải pháp sẵn có

(còn tiếp)

* 1. **Tổng quan SOA**
* BPM, EA và OOAD là những nguyên tắc mô hình hoá có quan hệ độc lập với nhau
* SOA hướng tới cung cấp các giải pháp nghiệp vụ của doanh nghiệp mở rộng hay thay đổi theo nhu cầu
* Giải pháp SOA là sự kết hợp các service có khả năng sử dụng lại với các interface chuẩn, được công bố và định nghĩa tốt.
* SOA cung cấp cơ cấu tương tác các ứng dụng sẵn có mà không quan tâm đến nền tảng và ngôn ngữ của ứng dụng.
* Có 3 level chính của sự trừu tượng hoá trong SOA
  + *Operations*
  + *Services*
  + *Business Processes*
  1. **Các bước của cách tiếp cận SOA**



1. Demain decomposition
2. Goal – service model creation
3. Subsystem analysis
4. Service allocation
5. Component specification
6. Structure components and services using patterns
7. Technology realization mapping
   1. **Service Contract**
   2. **Service Coupling**

* Giảm sự phụ thuộc vào các service khác mà bản thân service gọi để sử dụng.
* Giảm sự phụ thuộc vào chương trình sử dụng bản thân service.
* Trước đây, nhiều ứng dụng phát triển có sự gắn kết chặt với môi trường lập trình và chung quanh kiến trúc công nghệ.
* Thường gặp hơn nữa là sự gắn kết giữa ứng dụng và các thành phần của nó.
* Nguyên lý nhấn mạnh việc giảm kết dính giữa các thành phần của một service. Đặc biệt là giảm độ kết dính giữa:
  + service contract và chương trình sử dụng.
  + service contract và phần thực thi bên dưới của service.
* Mức độ kết dính chính là mức độ phụ thuộc.
* Kết dính phần mềm thể hiện một kết nối hay mối liên quan giữa 2 chương trình hoặc 2 component

***Service Contract Coupling Types***

1. Logic-to-Contract Coupling
2. Contract-to-Logic Coupling
3. Contract-to-Technology Coupling
4. Contract-to-Implementation Coupling
5. Contract-to-Functional Coupling

***Service Consumer Coupling Types***

***Service Loose Coupling and Service Design***

* 1. **Service Abstraction:**

Là nguyên lý có tác dụng làm giảm consumer-to-contract coupling

* + 1. **Types of Meta Abstraction**
* **Technology Information** —Meta data mô tả thực thi thuộc về kỹ thuật của service logic bên dưới
* **Functional Information** — Meta data mô tả service có khả năng gì
* **Programmatic Logic Information** —Meta data mô tả cách thức dịch vụ thực hiện khả năng của nó
* **Quality of Service Information** — meta data mô tả service behavior, giới hạn, yêu cầu tương tác .
  + 1. **Technology Information Abstraction**
* Ần những chi tiết về kỹ thuật được dùng xây dựng phần mềm, để tạo sự tự do trong việc thay đổi về kỹ thuật mà không ảnh hưởng đến người dùng đang sử dụng
  + 1. **Functional Abstraction**
* Functional abstraction xác định những khả năng nào được publish ngang qua techinal contract của nó
  + 1. **Programmatic Logic Abstraction**
* Áp dụng ở những chi tiết thiết kế thấp như: thuật toán, xử lý biệt lệ, exception handling, logging routines hay những xử lý khác gắn với việc xây dựng chương trình như thế nào
  + 1. **Quality of Service Abstraction**
* Chất lượng của service data là một giới hạn của vùng thông tin liên quan đến độ tin cây, các rule, và các thông tin hành vi của 1 service
  + 1. **Service Abstraction and Service Design**
* Service Abstraction vs. Service Encapsulation
  + *Encapsulation*: xác định giới hạn chứa information, resources và logic
  + *Abstraction*: có liên quan đến những phần mà service bao đóng, phần nào được trình bày và những phần nào được ẩn đối với consumer programs bên ngoài giới hạn của service
* Service Abstraction and Non-Technical Contract Documents: Cần có sự cân nhắc khi mô tả tài liệu phi kỹ thuật của contract
* Service Abstraction and Granularity
  + Service abstraction khuyến khích chúng ta publish ít chi tiết hơn để tạo sự thoải mái, tự do cho việc mở rộng service.

→Điều này ảnh hưởng trực tiếp đến tính hột với khuynh hướng tính chất hột càng thô hơn

* Service Abstraction and Service Models
  + Service model ít tác động đến sự mở rộng việc trừu tượng nội dung
    1. **Service Abstraction and Standardized Service Contract**
* Các quy tắc chuẩn hoá thường được tập trung vào việc đặt tên, kiểu dữ liệu, biểu hiện chính sách, và cấu trúc contract tổng thể.
* Phạm vi chi tiết và các loại ràng buộc có thể bị ảnh hưởng bởi mức độ của trừu tượng công nghệ, chức năng, và logic mà chúng ta quyết định áp dụng.

→Service Abstract thường kết thúc việc định hình nội dung contract một cách gián tiếp thông qua việc ảnh hưởng đến các tiêu chuẩn thiết kế

* 1. **Service Reusability**
  2. **Service Autonomy:**
* Trong lĩnh vực phần mềm, autonomy thể hiện sự độc lập của một chương trình thực thi với xử lý của nó
* Hai lợi ích chính mà tính autonomy mang lại là tăng khả năng dự đoán và khả năng tin cậy của chương trình
* Autonomy hỗ trợ khả năng sử dụng lại và khả năng kết hợp các service
* Việc gia tăng service autonomy đòi hỏi các yêu cầu cơ sở hạ tầng đáng kể

Các loại **Service Autonomy:**

* **Runtime Autonomy (execution)**
* Mức độ điều khiển của một service trong quá trình xử lý của nó tại thời điểm mà service đó được gọi và thực thi được gọi là runtime autonomy
* Mục đích của việc gia tăng runtime autonomy để đảm bảo những điều sau đây để phục vụ người dùng:
  + Chất lượng (hiệu suất) thực thi runtime có thể chấp nhận được
  + Mức độ tin cậy của sự thực thi cao hơn
  + Ở mức độ cao hơn là mức độ dự đoán hành vi (đặc biệt khi truy cập đồng thời)
  + Một composition tồn tại như một tập hợp các chương trình (services). Và nó có thể tham gia vào trong các composition khác 🡪 khuynh hướng không tự quản.
  + Tính tự quản của một service composition được quyết định bởi tập hợp sự tự quản của tập hợp các service của service composition đó
* **Design-Time Autonomy (governance)**
* Nhiều service consumer dùng một service sẽ tạo ra những sự phụ thuộc về design-time. Điều này hạn chế khả năng phát triển một service để đáp ứng các yêu cầu thay đổi trong tương lai. **Mức độ tự do của chủ service để tạo sự thay đổi đối với 1 service trong thời gian tồn tại của nó được xem như là design-time autonomy**.
* Sự trói buộc của chương trình người dùng và nghĩa vụ của các service vào contract của service đó🡪 giảm mức độ điều khiển trên sự phát triển service
* Có nhiều khía cạnh cần thiết của SOA, để đạt được chất lượng Design-Time Autonomy:
  + Khả năng để chia một service để đáp ứng những nhu cầu sử dụng ở mức cao hơn
  + Tuỳ chọn để sửa đổi hay gia tăng service’s hosting environment
  + Tự do bổ sung, nâng cấp hay thay thế công nghệ của một service trong việc đáp ứng những yêu cầu mới hay đáp ứng nhu cầu những cách tân service
* **Tăng sự tự quản trong thực thi (runtime autonomy) sẽ làm gia tăng design-time autonomy và ngược lại**
  1. **Service Statelessness**
  2. **Service Discoverability**
  3. **Service Composability**

1. **THIẾT KẾ CHƯƠNG TRÌNH**
   1. **Đặc tả các chức năng**
      1. **Đăng ký**

Người dùng chưa có account có thể đăng ký một tài khoản để sử dụng được những chức năng dành riêng cho thành viên. Người dùng sau khi chọn chức năng đăng ký, người dùng phải nhập đầy đủ thông tin yêu cầu.

***Các hàm OperationContract:***

* *DangKy():* có nhiệm vụ kiểm tra lại thông tin nhập nhập trên giao diện và trong cơ sở dữ liệu. Nếu có sai sót về thông tin nhập hay tài khoản đã tồn tại trong cơ sở dữ liệu thì trả về client thông báo yêu cầu người dùng nhập lại. Nếu không thì lưu thông tin đăng ký vào cơ sở dữ liệu và trả về client thông báo đăng ký thành công.
* *MaHoa():* có nhiệm vụ mã hóa mật khẩu người dùng đăng ký và lưu vào cơ sở dữ liệu
  + 1. **Đăng nhập**

Để có sử dụng các chức năng dành riêng cho các thành viên, người dùng cần đăng nhập vào account của mình. Người dùng cần nhập đầy đủ và chính xác thông tin username và password tài khoản của mình.

***Các hàm OperationContract:***

* *CheckAccount():* có nhiệm vụ kiểm tra username và password người dùng nhập với dữ liệu trong cơ sở dữ liệu. Nếu đúng thì trả về client thông báo thành công, ngược lại thông báo yêu cầu người dùng nhập lại.
  + 1. **Đăng xuất**

Khi người dùng không sử dụng các dịch vụ của hệ thống thông tin giao thông, có thể dùng chức năng này đăng xuất tài khoản đăng nhập của mình, tránh trường hợp mất mát tài khoản.

* + 1. **Danh sách địa điểm (đường) cần lưu ý**

Chức năng này liệt kê cho người dùng các địa điểm cần lưu ý khi lưu thông trên đường như: tìm địa điểm (đường) thường kẹt xe, địa điểm (đường) thường xảy ra nạn rải đinh, địa điểm (đường) thường xảy ra tai nạn giao thông, địa điểm (đường) có lô-cốt, … Người dùng có thể dò tìm trong danh sách các địa điểm (đường) cần lưu ý hoặc nhập thông tin địa điểm (con đường) và chọn tiêu chí cần kiểm tra để kiểm tra địa điểm (con đường) đó có thuộc danh sách các địa điểm cần lưu ý hay không.

***Các hàm OperationContract:***

* *Show\_DiaDiem\_LuuY():* sẽ liệt kê danh sách các địa điểm cần lưu ý.
* *Check\_DiaDiem\_LuuY():* có nhiệm vụ kiểm tra thông tin nhập của người dùng với danh sách các địa điểm cần lưu ý. Nếu tồn tại thì service trả về client thông báo tìm thấy và thông tin đầy đủ về địa điểm (con đường) đó như tên địa điểm (con đường), tình trạng, thời gian xảy ra (đối với các địa điểm hay đường hay kẹt xe, hay xảy ra tai nạn giao thông). Ngược lại, sẽ thông báo địa điểm (con đường) đó không thuộc danh sách các địa điểm cần lưu ý.
  + 1. **Tìm kiếm thông tin đường**

Chức năng này hỗ trợ cho người dùng tìm kiếm thông tin một con đường với các tiêu chí khác nhau: địa điểm giải trí, bệnh viện, trường học, tiệm bánh, ngân hàng, cây xăng, … hay chỉ đơn thuần là tìm kiếm thông tin của con đường. Ví dụ người dùng muốn tìm các địa điểm giải trí trên đường CMT8, người dùng chỉ việc nhập tên đường CMT8 và chọn tiêu chí tìm kiếm là giải trí, hệ thống sẽ tìm kiếm thông tin và trả lại kết quả cho người dùng.

***Các hàm OperationContract:***

* *Thong\_Tin\_Duong():* kiểm tra tên đường người dùng nhập với dữ liệu trong cơ sở dữ liệu. Nếu không tồn tại thì thống báo cho người dùng và yêu cầu nhập lại. Ngược lại sẽ tiếp tục tìm kiếm thông tin con đường với tiêu chí người dùng cần. Sau đó service sẽ trả về client kết quả tìm được, client sẽ hiển thị thông tin tìm được ra cho người dùng.
  + 1. **Tìm kiếm thông tin địa điểm**

Chức năng này cho phép người dùng có thể biết được đầy đủ thông tin về địa điểm cần tìm kiếm. Người dùng chỉ việc nhập thông tin địa điểm cần tìm, hệ thống sẽ tìm kiếm trong cơ sở dữ liệu và trả về các thông tin địa điểm đó như: tên địa điểm, loại địa điểm (thuộc loại địa điểm giải trí, trường học, ngân hàng, hay là địa điểm du lịch, …), tình trạng của địa điểm (có thuộc danh sách các địa điểm cần lưu ý hay không?), giờ hoạt động (nếu có).

***Các hàm OperationContract:***

* *Thong\_Tin\_DiaDiem():* có nhiệm vụ kiểm tra thông tin nhập với dữ liệu trong cơ sở dữ liệu. Nếu tồn tại thì service sẽ trả về client kết quả tìm được, và client sẽ hiển thị các thông tin cần thiết cho người dùng.
* *Check\_DiaDiem\_LuuY():* hàm được dùng lại trong quá trình thực thi tìm kiếm thông tin địa điểm.

Đặc biệt nếu địa điểm đó là các bến xe như bến xe miền Đông, bến xe miền Tây, … hay các sân bay thì ngoài hiển thị các thông tin ở trên, người dùng còn có thể biết được thông tin các chuyến xe, chuyến bay như ngày giờ khỏi hành, giá vé, chuyến đi, …

***Các hàm OperationContract:***

* *Thong\_Tin\_Chuyen\_Di():* dùng để tìm kiếm thông tin các chuyến bay hay chuyến xe của địa điểm đó.
  + 1. **Tìm kiếm đường đi từ điểm A tới điểm B**

Với chức năng này người dùng có thể tìm mọi đường đi từ một điểm A tới điểm B khác hoặc tìm đường đi giữa 2 địa điểm theo loại phương tiện di chuyển khác nhau như máy bay, tàu thủy, xe, tàu lửa, …

Người dùng chỉ việc nhập thông tin hai địa điểm. Hệ thống sẽ kiểm tra thông tin nhập với dữ liệu trong cơ sở dữ liệu và tìm kiếm đường đi giữa hai địa điểm đó theo loại phương tiện di chuyển. Người dùng có thể nhấn nút *Search* trong quá trình nhập địa điểm để có thể chọn được địa điểm chính xác như mong muốn.

***Các hàm OperationContract:***

* *TimDiaDiem():* dùng để tìm kiếm thông tin về địa điểm người dùng nhập khi người dùng nhấn nút *Search*. Service sẽ gọi hàm này để lấy các thông tin ở trong cơ sở dữ liệu, sau đó trả về client thông tin tìm được và hiển thị ra cho người dùng.
* *TimDuongDi():*sẽ lấy thông tin hai địa điểm A và B, sau đó tìm đường đi giữa hai địa điểm đó. Service trả về các kết quả tìm được cho client, và client sẽ hiển thị kết quả đó ra màn hình.
* *TimDuongDi\_PhuongTien():*đầu tiên sẽ gọi lại hàm *TimDuongDi()* để tìm toàn bộ các đường đi giữa hai điểm, sau đó sẽ tìm kiếm đường đi theo phương tiện di chuyển trong các kết quả vừa tìm được. Kết quả cuối cùng service sẽ trả về client, client sẽ hiển thị kết quả đó ra màn hình. Ngược lại nếu không tìm được, client sẽ hiển thị thông báo.

Ngoài ra người dùng còn có thể tìm đường đi với các tiêu chí khác như tránh đường có lô-cốt, kẹt xe, tránh đường hẻm, … . Ví dụ như muốn tìm đường đi từ CMT8 tới Lạc Long Quân với yêu cầu tránh các đoạn đường hay kẹt xe thì người dùng chỉ cần nhập hai địa điểm trên vào ô tìm kiếm, sau đó chọn tiêu chí tìm kiếm là tránh kẹt xe. Hệ thống sẽ tìm kiếm trong cơ sở dữ liệu và trả kết quả tìm được về client. Nếu tìm được đường đi thích hợp, service trả kết quả về client và client sẽ hiển thị kết quả ra màn hình. Nếu không tìm được đường đi nào thỏa yêu cầu tìm kiếm, service vẫn sẽ trả về các kết quả đường đi giữa hai điểm cho client, client sẽ hiển thị thông báo không tìm thấy đường thích hợp và hiển thị các kết quả khác như là một gợi ý cho người dùng.

***Các hàm OperationContract:*** Ngoài các hàm ***OperationContract*** kể trên, ta sẽ dùng thêm một số hàm ***OperationContract*** khác như:

* *TimDuongDi()*
* *Check\_DiaDiem\_LuuY()*: sẽ được sử dụng để so sánh các kết quả tìm được của hàm *TimDuongDi()*với danh sách các địa điểm cần lưu ý thỏa điều kiện tìm kiếm.
  + 1. **Tìm đường đi ngắn nhất giữa hai địa điểm**

Khi sử dụng chức năng này người dùng có thể tìm đường đi có khoảng cách ngắn nhất giữa hai điểm nhập vào. Thao tác thực hiện chức năng cũng giống với thao tác thực hiện chức năng tìm đường đi giữa hai điểm.

***Các hàm OperationContract:***

* *TimDuongDi()*
* *Short\_Path():* tính toán khoảng cách giữa các điểm trong kết quả tìm được của hàm *TimDuongDi()*, sau đó tìm ra cách đi từ điểm A tới điểm B sao cho khoảng cách là ngắn nhất.
  + 1. **Danh sách địa điểm yêu thích**

Chức năng chỉ dành cho thành viên có account. Khi tìm kiếm một địa điểm nào đó, người dùng có thể lưu địa điểm đó vào danh sách địa điểm yêu thích của mình. Người dùng chỉ cần chọn nút Địa điểm yêu thích để service sẽ lưu địa điểm đó vào cơ sở dữ liệu. Sau khi đã lưu xong, serivce sẽ trả về client thông báo.

***Các hàm OperationContract:***

* *DiaDiem\_YeuThich():* sẽ lưu địa điểm người dùng chọn vào cơ sỡ dữ liệu.
  + 1. **Đánh giá địa điểm (Rating)**

Chức năng chỉ dành cho thành viên có account, cho phép người dùng đánh giá một địa điểm nào đó. Người dùng sẽ đánh giá một địa điểm bằng cách chọn số sao từ 1 tới 5 trên giao diện. Mỗi account chỉ được đánh giá một lần. Mỗi lần đánh giá xong thì ***Rating*** của địa điểm đó sẽ tăng thêm điểm tương ứng với số sao của người dùng chọn. Hệ thống sẽ cập nhật ***Rating*** từng địa điểm thường xuyên.

***Ví dụ*:** địa điểm Diamond Plaza đang có số điểm Rating là 10 điểm. Một người dùng đánh giá địa điểm đó được 4 sao thì điểm Rating lúc đó sẽ là 14 điểm.

***Các hàm OperationContract:***

* *Update\_Rating():* hàm được dùng để cập nhật điểm Rating của từng địa điểm thường xuyên.
  + 1. **Danh sách các địa điểm yêu thích nhất**

Với chức năng này hệ thống sẽ hiển thị cho người dùng biết các địa điểm được yêu thích nhất. Số lượng địa điểm được yêu thích nhất hiển thị trên giao diện sẽ do người dùng chọn.

Ví dụ: Một khách hàng nào đó muốn xem 10 địa điểm được yêu thích nhất thì sẽ lần lượt thực hiện các thao tác chọn chức năng danh sách địa điểm yêu thích nhất, chọn số lượng kết quả hiển thị (top 10, top 50, top 100, …) là top 10. Sau khi chọn xong service sẽ thực hiện quá trình tìm kiếm và trả kết quả về client, rồi client sẽ hiển thị ra lại giao diện.

***Các hàm OperationContract:***

* *Show\_DiaDiem\_YeuThich():* sẽ thực hiện chức năng tìm kiếm các địa điểm có độ ***Rating*** cao nhất

Ngoài ra người dùng còn có thể kiểm tra xem một địa điểm bất kì có thuộc danh sách các địa điểm được yêu thích nhất vừa tìm kiếm hay không?

***Các hàm OperationContract:***

* *Check\_DiaDiem\_YeuThich():* sẽ thực hiện kiểm tra địa điểm đó với các địa điểm trong danh sách kết quả vừa tìm được.
  1. **Cơ sở dữ liệu**

1. **HIỆN THỰC**

# KẾT LUẬN

# TÀI LIỆU THAM KHẢO